"Information Technology and the Autonomous Control of a Mars In-Situ Propellant Production System"

by

Anthony R. Gross, K.R.Sridhar, William E. Larson, Daniel J. Clancy, Charles Peschur, and Geoffrey A. Briggs,
NASA-Ames Research Center, Moffett Field, CA 94035, USA

Abstract

With the rapidly increasing performance of information technology, i.e., computer hardware and software systems, as well as networks and communication systems, a new capability is being developed that holds the clear promise of greatly increased exploration capability, along with dramatically reduced design, development, and operating costs. These new intelligent systems technologies, utilizing knowledge-based software and very high performance computer systems, will provide new design and development tools, scheduling mechanisms, and vehicle and system health monitoring capabilities. In addition, specific technologies such as neural nets will provide a degree of machine intelligence and associated autonomy which has previously been unavailable to the mission and spacecraft designer and to the system operator.

One of the most promising applications of these new information technologies is to the area of in-situ resource utilization. Useful resources such as oxygen, compressed carbon dioxide, water, methane, and buffer gases can be extracted and/or generated from planetary atmospheres, such as the Martian atmosphere. These products, when used for propulsion and life-support needs can provide significant savings in the launch mass and costs for both robotic and crewed missions. In the longer term the utilization of indigenous resources is an enabling technology that is vital to sustaining long duration human presence on Mars.

This paper will present the concepts that are currently under investigation and development for mining the Martian atmosphere, such as temperature-swing adsorption, zirconia electrolysis etc., to create propellants and life-support materials. This description will be followed by an analysis of the information technology and control needs for the reliable and autonomous operation of such processing plants in a fault tolerant manner, as well as the approach being taken for the development of the controlling software. Finally, there will be a brief discussion of the verification and validation process so crucial to the implementation of mission-critical software.

Introduction

With exponentially increasing capabilities of computer hardware and software, including networks and communication systems, a new balance of work is being developed between humans and machines. This new balance holds the promise of greatly increased space exploration capability, along with dramatically reduced design, development, test, and operating costs. New information technologies, which take advantage of knowledge-based software and high performance computer systems, will enable the development of a new generation of design and development tools, schedulers, and vehicle and system health monitoring capabilities. Such tools will provide a degree of machine intelligence and associated autonomy which has previously been unavailable to the mission and spacecraft designer and to the system operator.

For budgetary reasons, a critical challenge for NASA today is to accomplish more space exploration for fewer dollars. Higher levels of machine intelligence and autonomy will be one key to the success of this strategy.

One of the current strategies actively being considered for the human exploration of Mars is the in-situ generation of propellant and lifesupport gases from the Martian atmosphere. There are a number of well-understood chemical processes by which this can be accomplished, and it then becomes an engineering consideration as to which process best suits the mission objectives and constraints.

To date NASA has used ground controllers to monitor the health of space systems and to control their activities. This is inherently costly, and new technologies must be applied to reduce this cost while maintaining the same degree of safety and control. Additionally, an in-situ propellant production (ISPP) plant on Mars will experience significant communications lag time and blackouts that make the control of such a system untenable from the Earth [R1,R2].

Traditionally, autonomous control of a complex physical device or system is achieved by developing software that is designed to respond appropriately to a predefined set of failures [R3]. To accomplish this task, the software developer must think through the global interactions that occur in the system to determine how a given failure will be detected and what is the appropriate recovery. As a device is expected to operate autonomously for an extended period of time, however, this task becomes increasingly difficult due to the complex interactions that may result from multiple failures. Thus, software developed using this approach tends to be costly to develop, hard to maintain, and limited in its ability to respond in a novel circumstance.

Researchers and engineers within NASA are addressing these challenges by developing a set of model-based reasoning techniques that are revolutionizing the rapid deployment of highly capable software by providing the ability to write programs through high-level compositional models.

[R43,R510] These models use a qualitative representation to describe the behavior of individual components within the system under both nominal and failure conditions. The component-level models are then combined into a global model that describes the interactions between components thus allowing an inference engine to reason generatively about the behavior of the device under various operating conditions. The expected behavior of the device is compared against the available observations to detect and isolate a failure and to select an appropriate response. This technology will allow an ISPP plant to operate untended on the Martian surface for the 500 days required by current human mission plans. Model-Based Reasoning technology is currently favored for application to Mars In-Situ Propellant Production and for robotic and human missions.

In the next section we will describe the key concepts currently under consideration for a Mars ISPP plant.

CHEMICAL PROCESSES FOR IN SITU RESOURCE UTILIZATION ON MARS:

The processes studied for production of propellant, oxygen and water on Mars are the following:

1. Solid oxide electrolysis
2. Sabatier reactor
3. Reverse water gas shift reactor

Solid oxide electrolysis:
This process generates oxygen from the predominantly carbon dioxide atmosphere of Mars using solid oxide electrochemical cells. An oxygen ion conductor such as yttria-stabilized zirçonia (YSZ) electrolyte is sandwiched between porous electrodes, platinum for example, to form an electrolysis cell. Carbon dioxide is split to carbon monoxide and oxygen and the oxygen is pumped electrochemically from the cathode to the anode. This endothermic net cell reaction is as follows:

$2CO_2 = 2CO + O_2$ _H= 566 kJ/ mol, _G = 514 kJ/ mol (1)

The solid oxide electrolysis approach has the advantages of producing 100 percent pure oxygen since the transport process in the electrolyte is solid state. Unlike the Sabatier process, oxygen is produced from the atmosphere without the need for any consumable or intermediary raw materials that are brought from Earth. This process produces oxygen and carbon monoxide in the proper stoichiometric ratio for a rocket. Initial design calculations by NASA for a Mars sample return missions indicated that the low specific impulse offered by a CO/ O_2 rocket does not warrant its use for ascent vehicle propulsion. However, recent works – that of Diane Linne at the NASA Glenn Research Center and Orbitec's solid CO / LOX rocket, seem to indicate that a CO/ O_2 rocket may indeed be suitable for a Mars ascent vehicle.

Detailed descriptions of the solid oxide electrolysis process can be found elsewhere (Sridhar and Vaniman, 1996, Sridhar 1995). A brief summary of the principle of operation is provided here. The

oxygen generator works on the principle of solid oxide electrolysis. At elevated temperatures, solid oxide electrolytes such as yttria stabilized zirconia become oxygen ion conductors. The basic configuration of the electrolysis cell is shown in Figure 1. At the cathode, CO_2 dissociates to form CO and O. The oxygen atom reacts with the incoming electrons from the external circuit to form an oxygen ion. The oxygen ion is conducted through the vacancies in the crystal structure of the electrolyte to the anode. At the anode, the oxygen ion donates the electrons to the external circuit to form an oxygen atom. Two oxygen atoms combine to form an oxygen molecule at the anode side of the cell.

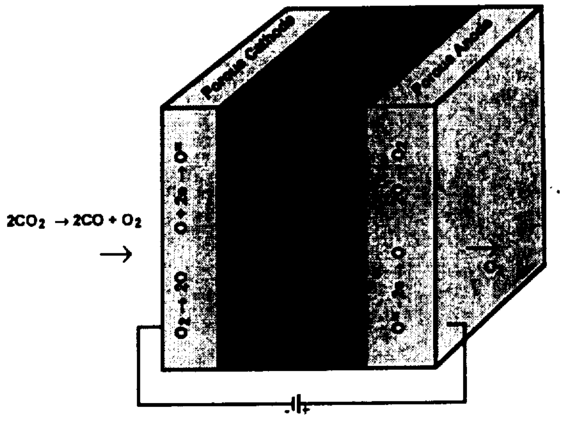


Figure 1. Principle of Operation of a Solid Oxide Electrolyzer.

Figure 2 shows a simple schematic of an oxygen generation plant that utilizes the solid oxide electrolyzer technology. Since the Mars atmosphere is at 8 hPa ambient pressure, a front end compressor is used to get enough throughput in the electrolyzer.

Figure 2. Simplified Oxygen Generation Plant (OGS) Flow Schematic.

An oxygen generator based on this technology is manifested to fly on the 2001 Mars Surveyor Lander. Figure 3 is a photograph of the engineering model of the flight unit hardware.

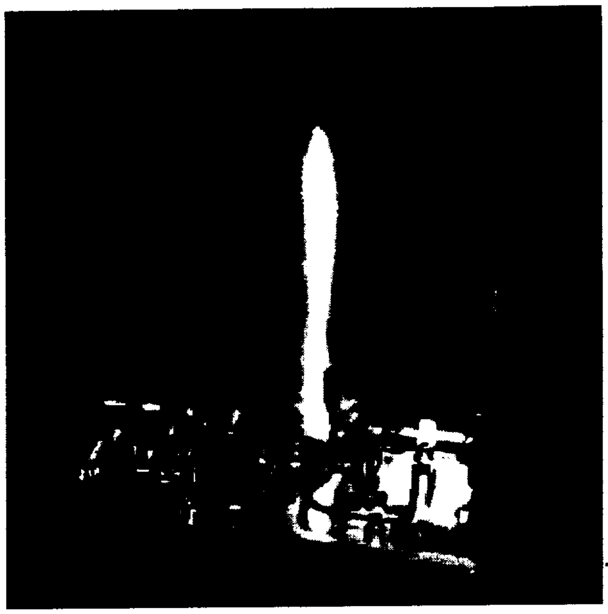


Figure 3. A Photograph of the OGS in the Development Unit Configuration.

Sabatier / Water Electrolysis:

The other prime candidate technology for the Mars in-situ propellant production is the Sabatier electrolysis (SE) system. The subsystem components of the SE system have been well known to the chemical industry for a long time. Space-qualified components for the subsystems have been available for some time through DoD and NASA funded programs mainly for closed loop and/or regenerative life support systems. Such systems have been developed by Hamilton Standard in the sixties and later by Allied Signal, Boeing, and Dornier. There is a rich heritage of such systems from the former Soviet space program also. Using such systems for Mars propellant manufacture was first suggested by Ash et al in his seminal 1976 paper. Experimental work on integrated SE systems designed for Mars propellant manufacture began in 1993, with funding support from the New Initiatives Office at NASA JSC, a full scale (for a MSR mission application) working unit was built by Zubrin, Steve Price, and Larry Clark at Martin Marietta Astronautics (now Lockheed Martin Astronautics) in Denver.

Carbon dioxide acquired from the Martian atmosphere is reacted with hydrogen in accord with reaction (2)

$$4H_2 + CO_2 = CH_4 + 2H_2O \qquad DH= -40 \text{ kcal/mole} \qquad (2)$$

Reaction (1), known as the "Sabatier reaction," is highly exothermic and has a large equilibrium constant ($\sim 10^9$) driving it to the right. It occurs spontaneously in the presence of either a nickel or ruthenium catalyst (nickel is cheaper, ruthenium is better) at temperatures above 250 C. (Typical reactors operate with peak temperatures around 400 C in the forward reaction zone, declining to

200 C at the exit). The methane and water produced by reaction (2) are easily separated in a condenser. The methane is then liquefied and stored, while the water is electrolyzed in accord with:

$2H_2O = 2H_2 + O_2$ DH= +57 kcal/mole (3)

The oxygen so produced is liquefied and stored, while the hydrogen is recycled back into the Sabatier reactor to produce more methane and water, and so forth.

The primary disadvantage of the SE system is the need to import hydrogen. This requirement is especially difficult on the MSR mission, where the relatively small tank sizes employed increases the tank surface area/volume ratio, increasing heat-leak and thus boiloff, making transport of the required hydrogen to Mars difficult. The SE process, operating alone, produces 2 kg of oxygen for every one kg of methane. But the optimal mixture ratio to burn O_2/CH_4 in a rocket engine is not 2/1 but about 3.5/1, where an engine specific impulse as high as 380 s can be achieved. If oxygen is also required for life support, then the ratio has to be greater than 3.5/ 1.0. If the SE process is acting alone, the only way to achieve this mixture ratio is to throw away some of the methane produced. This is undesirable due to the cost of carrying hydrogen from Earth and also the power required for propellant processing.

Reverse water gas shift reactor:

The reverse water gas shift (RWGS) reaction has been known to chemistry since the mid 1800's. While it has been discussed as a potential technique for Mars propellant manufacture in the literature, there is very little experimental work done to demonstrate its viability for such application to-date. The RWGS reaction is given by equation (4).

$CO_2 + H_2 = CO + H_2O$ DH= +9 kcal/mole (4)

This reaction is mildly endothermic and will occur rapidly in the presence of an iron-chrome catalyst at temperatures of 400 C or greater. Unfortunately, at 400 C the equilibrium constant Kp driving it to the right is only about 0.1, and even at much higher temperatures Kp remains of order unity. There is thus a significant problem in driving the RWGS reaction to completion.

The proponents of this technology have claimed that RWGS is the solution to the hydrogen imbalance in the SWE process. Their claim, that is not valid, is: reaction (4) can be driven as written, then an "infinite leverage oxygen machine" can be created by simply tying reaction (4) in tandem with the water electrolysis reaction (3). That is, the CO produced by reaction (4) is discarded while the water is electrolyzed to produce oxygen (the net product), and hydrogen which can be recycled to reduce more CO_2. Since all the hydrogen is recycled, barring leakage losses this can go on forever allowing the system to produce as much oxygen as desired. The only imported reagent needed is a small amount of water which is endlessly recycled.

The practical difficulties of the RWGS scheme are discussed here. It is important to note that the H_2O shift reaction is very effective in the forward direction,

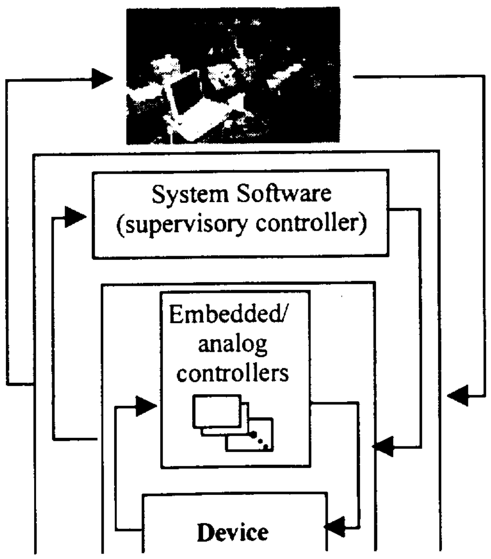
$CO + H_2O = CO_2 + H_2$

It is exothermic with a 99% CO conversion in a single pass. Reversing the H_2O shift reaction requires special catalysts and temperature controls to prevent the endo-thermic reaction from reverting to the forward direction. A typical single-pass conversion of CO_2 and H_2 would be in the range of 10%.To obtain a conversion in the 90% range would require multiple cycles. While conversion rates could improve somewhat if the products could be separated and removed from the hot zone of the reactor, this separation and removal seem very difficult to achieve. Because of the low single-pass conversion, the typical reactor output would include a mixture of H_2, CO_2, CO, and H_2O. The H_2 and H_2O must be recovered to conserve the Earth-sourced H_2. The CO must be separated from the CO_2 to be recycled in order for the subsequent pass to reach the 10% conversion. The recovery of H_2O by the near-freezing condensation is quite effective for single-pass systems. However, the multiple recycles of the RWGS reactor will add a significant quantity of residual H_2O being rejected with cold products that, if not recovered by other means, will impact the H_2 conservation. The membrane recovery of H_2 from the product stream is based on a partial pressure differential diffusion. Thus a vacuum pump is required for separation and even then a 10% loss would be expected in any practical design.

In the next section we will discuss concepts for autonomous control of such an in-situ propellant production plant, as well as an example of current practice.

Autonomous Control Concepts

The ability to autonomously monitor and control complex devices such as an ISPP plant is critical to NASA's ability to accomplish many of its long term exploration goals. From the beginning of the Space Program, control of spacecraft and systems have been largely managed by a large number of highly trained, ground control personnel. This has its roots in the limited capability of computers of that age and their massive size. Instead, sensor data was telemetered to the ground where a room full of systems experts would monitor each individual system's health and send commands to the spacecraft directly or via an Astronaut. Over the past forty years, we have seen a



radical shift in this paradigm due to the advent of computer technology. Automation eased the burden of the ground controller and the astronaut, but often the capabilities performed by the software is still quite rudimentary due to both computational resource limitations and the difficulty encountered when trying to develop, test and validate software that provides the required functionality. As we move outward in the solar system, beyond the Earth-Moon system, the realities of space exploration will require new control technologies due to both physical and fiscal constraints.

Conceptually, the task of controlling a device such as an ISPP is simply an attempt to maintain the system in a stable state while issuing commands that transition the device through a series of configurations designed to accomplish a sequence of goals or objectives in some optimal fashion. Accomplishing this task, however, is often quite difficult due to the normal variations that occur within both the process and the environment, limited observability into the current state of the device and the potential of abrupt failures and degraded component performance. Traditionally, these problems are solved through the use of a tiered architecture comprised of three levels (see figure 1):

1. *analog and embedded feedback controllers* to perform low-level regulatory functions,
2. *higher-level system software* to perform nominal command sequencing and threshold monitoring to detect and respond to off-nominal conditions, and
3. *humans* to generate the command sequences, monitor the state of the device to detect off-nominal conditions, diagnose failures when they occur and select recovery actions in response to these failures.

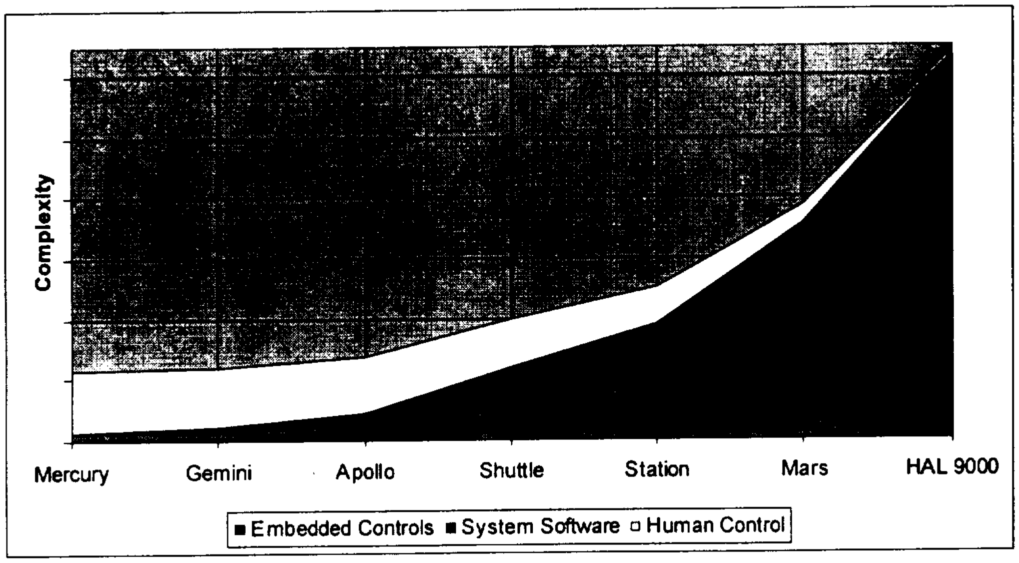


Figure 2: Progression across missions of the tasks performed by humans, system sof

While the capabilities provided by the system level software have increased drastically over the past 40 years, the complexity of the missions attempted by NASA has also increased. As a result, the requirements levied on the ground control team has often increased thus requiring larger ground support teams. As we look toward the future, this paradigm begins to break down due to both time-delay in the communication with Mars as well as the cost of maintaining such a large ground support team. As a result, the role traditionally performed by the ground support team is being shifted to the system level software thus drastically increasing the functionality required of this component. Figure 2 shows how the functionality provided by these three different components has shifted over the years and where we expect the responsibility to lie when supporting a manned expedition to Mars.

Currently, the system level software is developed by engineers that use their commonsense understanding of the hardware and mission goals to produce code and control sequences that will allow a spacecraft or other system to achieve some goal while allowing for some (usually very small) amount of uncertainty in the environment. In developing this code, the engineer must reason through complex sub-system interactions to generate procedural code that can account for all the different combinations of failures and off-nominal conditions that might occur. As the functionality that is required of the system-level software increases, development, test, validation and maintenance of this software using this traditional approach becomes quite difficult if not impossible due to the myriad of off-nominal conditions that the software is expected to handle. Furthermore, as the engineers gain a better understanding of how the device is behaving after deployment, it is often quite difficult to update the code to reflect the additional information that has been obtained.

Artificial intelligence and autonomous control

As we attempt to automate the processes that are traditionally performed by humans when monitoring and controlling these devices, it is clear that we will often need to replicate the sophisticated inferencing capabilities exhibited by humans when performing this task. Over the last 40 years, the field of artificial intelligence has been developing a variety of automated techniques that emulate a human's reasoning ability [14]. While the systems developed are far from performing at the visionary level exhibited by the HAL 9000 computer in *2001: A Space Odyssey,* recent accomplishments such as Deep Blue's victory over Kasparov have demonstrated that sophisticated inferencing tasks can be automated.

One of the most notable recent accomplishments within NASA is the development and demonstration of the Remote Agent (RA) autonomous control architecture as part of the Deep Space One mission within the New Millennium Program (NMP). The Remote Agent architecture, developed collaboratively between NASA Ames Research Center (ARC) and the Jet Propulsion Lab (JPL), combines high-level planning and scheduling, robust multi-threaded execution and model-based fault detection isolation and recovery into an integrated architecture that is able to robustly control a spacecraft over long periods of time [4, 11].

One of the primary components of the Remote Agent architecture is the Livingstone model-based health management system. Livingstone is an advanced inference engine that uses a high-level declarative model of the physical device to monitor the state of the device, detect off-nominal behavior, isolate failures to individual components and reason about alternative recovery actions.

The key benefit provided by Livingstone is the use of a first-principles model that describes the behavior of each component within the device and the interactions between the components [5,6,7]. By reasoning generatively about the behavior of the device using the model, Livingstone is able to detect failures whenever a discrepancy occurs between the observations and predictions. In addition, Livingstone is able to use the same model to generate the most likely hypothesis that is consistent with the observations and to select the optimal reconfiguration action for recovering from the failure. Thus, due to the use of a model of the device, Livingstone is able to reason about novel combinations of failures and avoids the need to develop mission specific code that must pre-enumerate all of the various failure combinations that might occur. Furthermore, the models used by Livingstone are easy to update and maintain and can often be reused across missions thus further reducing the software development costs while increasing the functionality provided.

The Livingstone model-based health management system

Recently, Ames Research Center and Kennedy Space Center have been investigating how the core ideas developed within the Livingstone system can be applied and extended to autonomously control an ISPP plant. In this section, we will provide a more detailed description of the Livingstone by explaining how these ideas are being applied to control of an ISPP plant.

Modeling Paradigm

As mentioned above, Livingstone uses a high-level, compositional model to identify the components within the device and the relationships between the components. This model is used for prediction, fault detection, isolation and reconfiguration. A Livingstone model is comprised of a set of components and connections between these components. Each component is modeled using a set of discrete valued variables. For example, a valve might be modeled using the variables *flow-in, flow-out, pressure-in* and *pressure-out* with values such as *zero, low, nominal, high.* For each component, a set of *modes* are defined identifying both the nominal and failure modes of the device. For each mode, a set of constraints are specified that restrict the values of the component variables whenever the component is in that mode. Thus, a valve might be modeled using modes such as *open, closed, stuck-open* and *stuck-closed* where the model of the valve in the *open* mode might be

$$\begin{gathered} \textit{flow-in} = \textit{flow-out} \\ \textit{pressure-in} = \textit{pressure-out} \end{gathered}$$

In addition, to the description of the behavior of the device for each mode, the model also includes transitions between modes with guard conditions describing when the transition occurs along with relative probabilities on the likelihood of the transitions. These probabilities are used to provide information about the relative likelihood of various failures. Figure 4 shows how a valve model might be represented as a finite-state automaton in which the labels on the links correspond to device commands.

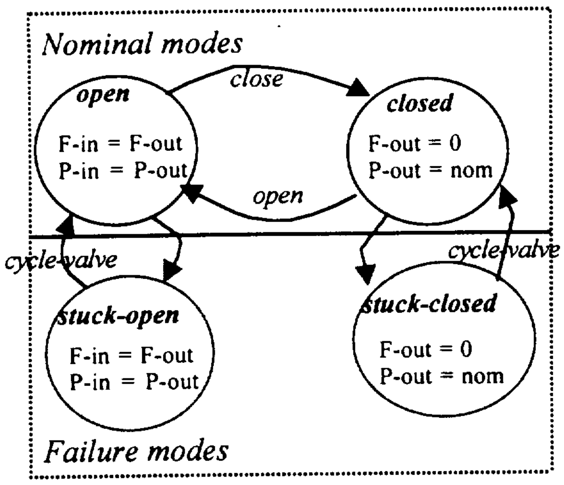


Figure 4: Valve model

One of the key benefits of this modeling paradigm is that the modeler is only responsible for describing the *local* behavior of each component and the relationships that exist between components. Livingstone then uses this specification to compose a larger, system model that can be used to reason about the *global* behavior of the entire system given the mode of each component. Furthermore, since the models are qualitative in nature it is often straight forward to develop these models many of these models even before the hardware design is complete.

Inference with Livingstone

Given a model of the form described in the preceeding section, Livingstone performs two main tasks: 1) inferring the current state of the device given the limited available sensor information; and 2) identifying an optimal set of commands for system reconfiguration following a failure or external perturbation that transitions the system out of the desired state. At first glance, it might appear that the valve model described in the previous section is too simple to be of much use in performing these tasks. In practice, however, we have found qualitative models of this nature quite effective for detecting a wide range of likely failures. In fact, it is exactly this type of models that humans often use when reasoning about the current state of a device.

To estimate the current state of the system, Livingstone monitors the sequence of discrete commands that are issued to the ISPP plant to track the expected state of the device and compares the predictions generated from its model against the observations received from the sensors. Once a discrepancy occurs, Livingstone performs diagnosis by searching for the most likely set of component *mode assignments*[1] that are consistent with the observations. This is done using a search technique called *conflict-directed, best-first search* developed within the model-based reasoning area of the artificial intelligence community. This search technique is able to efficiently search an exponentially large set of failure modes by focusing on the components whose state results in a conflict between the observations and the predictions. Within the Deep Space One experiment, this search technique was able to identify the most likely component failure within a couple hundred milliseconds for most device failures.

[1] The *state* of the device is represented by identifying the current *mode* of each component in the model.

Once the state of the system is identified, the same search techniques can then be used when reasoning about reconfiguration commands to identify the lowest cost set of commands that can be issued to transition the system into a state that satisfies the current operational goals provided by a higher level executive.

Model-based control of an ISPP plant

The current system architecture being developed to control an ISPP plant combines the Livingstone health management system with a real-time executive for commanding the device. Figure 5 shows a block diagram of this architecture. At the lowest level, embedded and analog controllers are used to perform low-level regulatory functions. Nominal commanding of the ISPP is performed by a real-time executive. As these commands are sent, the *Mode Identification* component of the Livingstone system monitors the commands to identify the expected state of the plant. The real-valued sensor data is processed by a set of *monitors* that abstract the real-valued information from each sensor into a set of a-priori defined discrete values such as *high, medium, low* or *plus, zero, minus*. When a failure occurs, the real-time executive is notified. For failures that require a very fast response time, the real-time executive might respond reactively in a predefined manner. For other failures, however; the executive requests a sequence of reconfiguration commands from the *Mode*

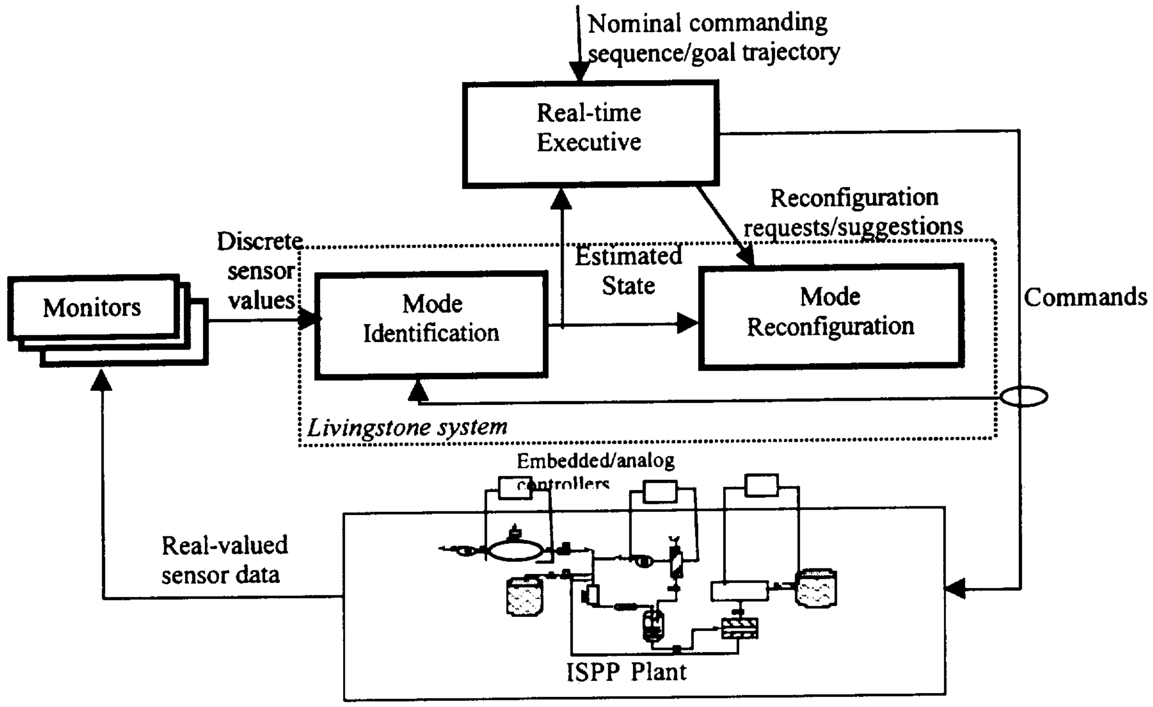


Figure 5: ISPP Control Architecture

Reconfiguration component of Livingstone and then continues commanding the device.

Demonstration Testbeds

To support the development and evaluation of this technology, we are currently developing both a hardware testbed and a simulation-based testbed. For the hardware demonstration, we are developing a testbed that uses Reverse Water Gas Shift (RWGS) to generate C0 and O_2 from the Martian atmosphere.

The RWGS reactor converts CO_2 and H_2 into CO and H_20 at a 10% efficiency rate. Thus, the outflow stream from the reactor contains liquid water, and gaseous CO, CO_2 and H_2. After exiting the reactor, a condenser is used to separate the water from the gases and then an electrolysis unit is used to separate the hydrogen from the oxygen. The oxygen is then stored while the hydrogen is fed back into the RWGS reactor. Similarly, the CO is extracted from the gas mixture and the remaining CO_2 and H_2 are routed back into the RWGS reactor. Control of an RWGS system is actually quite straightforward since the system has a limited number of components. A full-scale ISPP device, however, would require various other components along with redundant valves and flow controllers. As the number of components within the system increases, the probability of a failure increases and the discrete control problems become more complicated. The RWGS testbed, however, allows us to demonstrate how these techniques can be used to control a real physical device for an extended period of time.

At the same time, we are also developing a simulation testbed that uses a hypothetical ISPP flight article based upon a Sabatier/Electrolysis system for converting CO_2 and H_2 to CH_4 and O_2 coupled with a pair of zirconia cells for generating the extra O_2 that is required. . The design of the system for this simulation tries to balance many of the design considerations (e.g. mass and power limitations) that must be satisfied by a true flight article while also including redundant components and the additional margin that would be require to ensure a successful mission.

The simulation for the software testbed is based upon the *hybrid concurrent constraint (hcc)* programming language developed at ARC [12]. Hcc is a hybrid discrete/continuous modeling language that combines the benefits of a discrete event simulation with the precision provided by a dynamic simulation using ordinary differential equations. In addition, we are currently developing extensions to hcc that will allow us to perform a stochastic simulation that is able to inject faults, component degradation and variable process noise thus allowing us to test the control system under a broad range of conditions.

In addition to the demonstration of these techniques within the context of autonomous control of an ISPP plant, we are also applying these techniques to a variety of other missions. These missions include autonomous control of an advanced life support system and a space interferometer as well as integrated health management systems for two experimental reusable launch vehicles (X-34 and X-37).

Verification and Validation of Livingstone Models for ISPP

Verification of ADAC

Autonomous systems present difficult verification and validation (V&V) challenges. In contrast to conventional open-loop systems, they arbitrate many resources on-board using their own decision procedures. Because of this, the range of possible situations becomes very large and uncontrollable from the outside, making scenario-based testing much less inefficient.

Model checking is an analytical V&V technique based on exhaustive exploration of all possible executions of a (model of a) dynamic system. It provides a much better coverage than traditional testing, and can be applied at an earlier design stage, thus reducing the costs of repairing errors. Model checking is limited by state space explosion, however: the number of cases to be explored grows exponentially in the size of the system.

In collaboration with Carnegie Mellon University (CMU), NASA Ames is developing a translator that feeds Livingstone models into the SMV (what does SMV stand for?) model checker from CMU [P1]. SMV uses advanced symbolic computation techniques to represent and process huge state spaces in a compact and efficient way. The properties to be verified, expressed in a powerful temporal logic (CTL) (what does CTL stand for?) or using pre-defined specification patterns, are added along the Livingstone model and similarly processed by the translator. The translator thus enables model checking of Livingstone models by their developers in their Livingstone environment, without requiring them to use or learn the input language of the SMV tool.

This model checking technology will be used, along with traditional scenario-based technology, to validate the ADAC in the PUMPP experiment. (what does ADAC and PUMPP stand for? What is the PUMPP experiment?) It is already used by the ISPP Autonomous Controller team at the NASA Kennedy Space Center to support the development of their Livingstone model of ISPP. It is possible to find out, for example, whether the model allows a given configuration to happen. It takes less than a minute to SMV to answer this query, while symbolically analyzing a reachable space of the order of 10^{50} states!

Concluding Remarks

As we have seen, the requirements of implementing space exploration mission at lower cost, greater safety, and greater scientific return, along with new computer system capabilities to meet these requirements, have resulted in many new opportunities to expand human presence in the solar system. Recognizing the need to greatly expand research in the area of advanced computer system capabilities, NASA has embarked on the Intelligent Systems Program, which seeks to research and develop new capabilities in the areas of automated reasoning, human-centered computing, intelligent use of data, and concepts for revolutionary computing, such as biomemetics, nanotechnology, etc. Ultimately we will see a new balance of work between humans and intelligent machines, where tasks will reside with the entity having the best capability. This will provide the total human-machine system with the capability to explore far beyond the limits of the present.

References

References:
K. R. Sridhar, *J. Propulsion and Power*, **11**, 6 (1995).

Many of the following papers may be found on the World Wide Web at http://ic-www.arc.nasa.gov/ic/projects/mba/

[1] R. Zubrin and R. Wagner. The case for Mars: The plan to settle the Red Planet and why we must. The Free Press, 1996.

[2] S. J. Hoffman and D. I. Kaplan, editors. Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team. NASA Special Publication 6107. July 1997.

[3] B. C. Williams and P. Nayak, A Model-based Approach to Reactive Self-Configuring Systems, Proceedings of AAAI-96, 1996.

[4] D. E. Bernard et Al. Design of the Remote Agent Experiment for Spacecraft Autonomy. Proceedings of IEEE Aero-98.

[5] J. de Kleer and B. C. Williams, Diagnosing Multiple Faults, Artificial Intelligence, Vol 32, Number 1, 1987.

[6] J. de Kleer and B. C. Williams, Diagnosis With Behavioral Modes, Proceedings of IJCAI-89, 1989.

[7] J. de Kleer and B. C. Williams, *Artificial Intelligence*, Volume 51, Elsevier, 1991.

[8] D.Schreckenghost, M. Edeen, R. P. Bonasso, and J. Erickson. Intelligent control of product gas transfer for air revitalization.. Abstract submitted for 28th International Conference on Environmental Systems (ICES), July 1998.

[9] R. P. Bonasso , R.J. Firby, E. Gat, D. Kortenkamp, D. Miller and M. Slack. Experiences with an architecture for intelligent, reactive agents. In Journal of Experimental and Theoretical AI, 1997.

[10] B. C. Williams and P. P. Nayak. Immobile Robots: AI in the New Millennium. In AI Magazine, Fall 1996.

[11] N. Muscettola. HSTS: Integrating planning and scheduling. In Mark Fox and Monte Zweben, editors, Intelligent Scheduling. Morgan Kaufmann, 1994.

[12] V. Gupta, R. Jagadeesan, V. Saraswat. Computing with Continuous Change. Science of Computer Programming, 1997.

[13] G. M. Brown, D. E. Bernard and R. D. Rasmussen. Attitude and articulation control for the Cassini Spacecraft: A fault tolerance overview. In 14th AIAA/IEEE Digital Avionics Systems Conference, Cambridge, MA, November 1995.

[14] S. Russel and P. Norvig *Artificial Intelligence: A Modern Approach*, MIT Press 1995.

Bibliography

[P1] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J.